Transforming List

In [previous posts](http://data-structure-learning.blogspot.com/p/functional-programming-in-java.html) we saw functional interfaces, how to use them, iteration problems prior to Java 8 and how java 8 solved the iteration problem using forEach(..) method.

But let me remind you that forEach(..) method is not only used to iterate efficiently through the collection but also it can do certain activities inside it. forEach() method takes Consumer as input and performs action on it using accept method.

Let us say that we want to convert entire list of String to uppercase. So how will we do that. Well, we will iterate through the list by using [any of the iteration methods](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-6iterating-over.html) and use [toUpperCase(String str)](http://data-structure-learning.blogspot.com/2015/05/string-to-upper-case.html) to convert specified string to upper case.

Let’s say that we have list of computer languages.

**public** **static** List<String> populateDatabases() {

List<String> list = **new** ArrayList<String>();

list.add("Java");

list.add("C#");

list.add("MySQL");

list.add("MongoDB");

list.add("Oracle");

**return** list;

}

Now, we want to convert all the languages to new list. Below code does the specified job.

List<String> languages = *populateDatabases*();

**final** List<String> capsLanguages = **new** ArrayList<String>();

**for** (String language : languages) {

capsLanguages.add(language.toUpperCase());

}

Iterate through the list using enhanced forEach, take each element from the list and convert it to uppercase. Then add it to new list.

In previous post we saw how to iterate the list and print the elements using Standard println. For convenience I have written code below. Refer [this post](http://data-structure-learning.blogspot.com/2015/06/iterable-interfaces-foreach-method.html).

languages.forEach(language -> System.***out***.println(language));

In above statement for action we wrote System.***out***.println(language), Which means that our action was to print the language on to the screen.

Now let us change the action to convert the specified string to uppercase string.

languages.forEach(lan -> newLanguagesList.add(lan.toUpperCase()));

Output:

[JAVA, C#, MYSQL, MONGODB, ORACLE]

Now we saw that forEach is much more worth than iterating the collection.

Now let us run our forEach method on list that contains null.

**public** **static** List<String> populateDatabases() {

List<String> list = **new** ArrayList<String>();

list.add("Java");

list.add("C#");

list.add(**null**);//null value in list

list.add("MongoDB");

list.add("Oracle");

**return** list;

}

**final** List<String> newLanguagesList = **new** ArrayList<String>();

languages.forEach(lan -> newLanguagesList.add(lan.toUpperCase()));

Let’s run the above code. And we get

Exception in thread "main" java.lang.NullPointerException

at com.example.java8.fpinjava.chapter2.TransformingList.lambda$0(TransformingList.java:16)

at com.example.java8.fpinjava.chapter2.TransformingList$$Lambda$1/2055281021.accept(Unknown Source)

at java.util.ArrayList.forEach(Unknown Source)

at com.example.java8.fpinjava.chapter2.TransformingList.main(TransformingList.java:16)

Welcome the NullPointerException, most irritating and archenemy exception of Software Engineer.

How to escape null elements from list? Well I put null value in list intentionally to set next post in motion. Next post is about Predicate FunctionalInterface. It responds to a true or false for the given input.

So in this post we learnt how to transform list and forEach(..) method is not only for iteration.